V&V Tools

RPG Reference Document

11/30/00

Table of Contents

This document references specific tools and products. These are intended as examples only of the types of tools available in the M&S community and are not to be considered as endorsements. This document lists URL addresses that were current at the time of writing. They are intended to identify potential sources of information and no attempt has been made to maintain their currency. Given that any effective V&V process should be tailored to the specific problem or application being addressed, the selection of tools to be used in the performance of V&V activities must be based on the tasks to be performed and the techniques used to perform them.

This document corresponds to the web version of the VV&A RPG Reference Document of the same name and date. It has been modified for printing.

Introduction

This document provides information on some tools that can be used to perform V&V tasks. Use and selection of these tools is highly dependent on the criticality of the application, the maturity of the product, corporate culture, and the type of M&S development paradigm [Glasow and Pace, 1999].

The degree to which V&V tasks and activities can be automated directly impacts the efficiency of the overall V&V effort. As with any procedure requiring tools, it is important to select the correct tools for the job. Ideally, the M&S tools used in the development and/or preparation of the simulation should be highly integrated with the verification and support tools. Validation on the other hand, by its nature, is not as closely tied to the details of the M&S process.

In general, the entire computer aided software engineering (CASE) tools industry is available to the M&S and V&V community. However, for the purposes of V&V automation, tools that provide abstract design and analysis capabilities are of particular interest.

This is by no means meant to be an exhaustive discussion on software or M&S V&V tools. It is intended to provide a starting point for tool selection and to raise some basic issues with regard to tool usage during the V&V process.

Verification Tools

This section focuses on design/analysis and code testing tool classes and describes some appropriate commercial tools. The verification tool market is a large commercial market with an extensive offering. A list of more than 100 vendors is easily obtained from the World Wide Web (WWW). In many cases, a vendor offers several products ranging from single standalone utilities to complete Integrated Development Environments (IDEs).

The mainstay of the verification tools market is the design and analysis IDE, which can be enhanced by add-on utilities. High-performance tools are available for both the Windows and UNIX operating systems. IDE vendors offer attractive “turn-key” M&S tool solutions by adding functionality with an emphasis on ease of use. However, these products should be evaluated carefully. As in any highly competitive market, marketing may exceed functionality.

To achieve the goal of a verified model, the information in the conceptual model (e.g., statements of assumptions, algorithms, architectures) needs to be accurately transferred to the M&S development process. Design and analysis tools that operate at

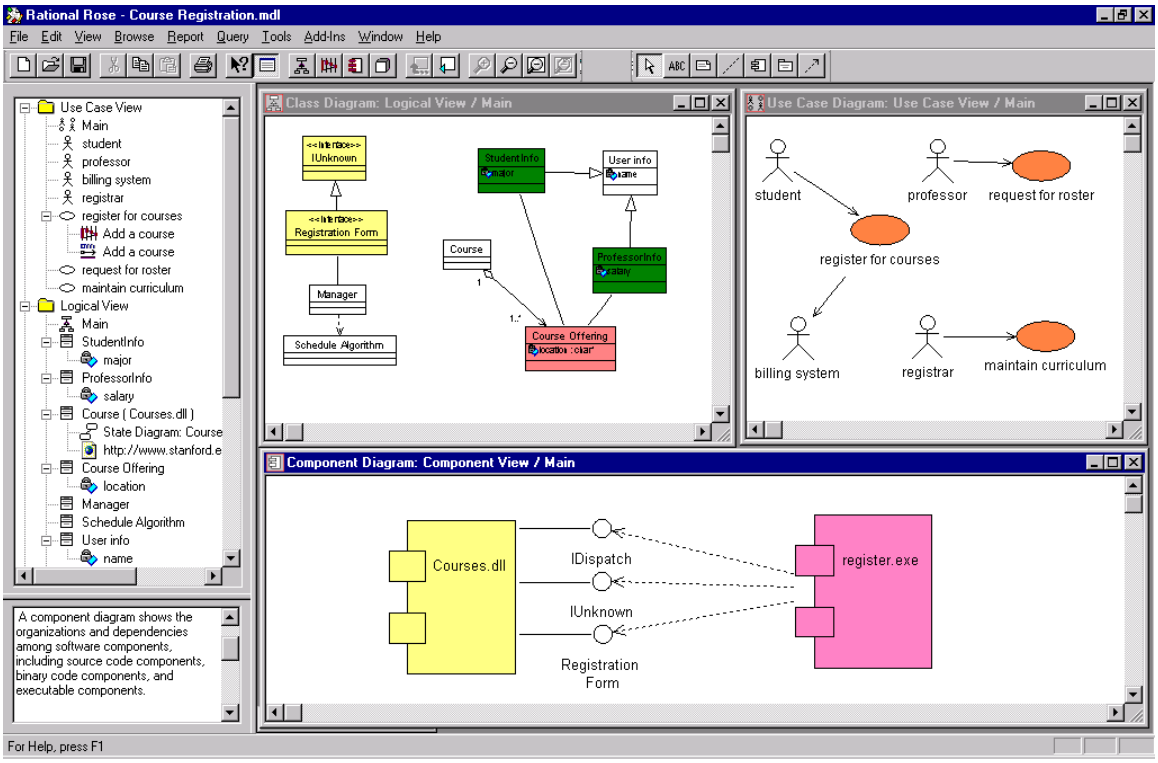
high levels of abstraction can facilitate this transfer of information by providing a link between concept and implementation details.

In addition to transferring the concept information into the M&S process, information from the M&S process needs to be extracted. Analysis and testing tools can automate much of this process by providing verification information about a model. Qualitative information such as code quality, portability, reusability, and run-time error analysis can be extracted. Quantitative data such as code metrics and intermediate test data provide valuable input to the verification process.

Classes of Verification Tools

Design and Design Analysis Tools

Modeling languages provide a means of describing the relationships of a model's processes. Graphical notation is used to depict a model's processes and their relationships without regard to implementation details. This inherently abstract view of the model allows the developer to approach the M&S development at the architecture level, which is closer to the conceptual model description. The figure below shows an example of modeling in Rational Rose, a popular verification product.



Rational Rose Sample Screen

A model represented in a modeling language can be translated into the developer's choice of coding languages directly by the tool. This approach to code development minimizes the errors in the overall architecture of the software and encourages good software engineering practices by providing a template for the developer to fill in the functional details.

Legacy models can be reverse-engineered to produce abstract representations that narrow the gap between implementation and specification. Models can either be reentered in a modeling language for analysis or, if directly supported, an analysis utility can produce a high-level depiction (e.g., a tree structure) of the model's internal relationships.

Low-level verification tools such as compilers, assemblers, and debuggers are the last step in the M&S development process. Even if a modeling language is used, it is assumed that the M&S process will eventually culminate in the generation of executable code. The error and warning reporting, as well as the low-level analysis capabilities of these tools, are the basic requirements for code verification.

Code Testing Tools

Code testing tools support quality analysis, automated loading, and metrics. The software engineering (SE) community has developed its own set of standards to facilitate the production of high-quality code. The use of utilities to exploit the existing knowledge base within the SE community will increase the probability of a high-quality M&S product.

Quality analysis tools give a qualitative report on how well the coding style follows good SE practices. By parsing the code, these tools report conditions where the code may function correctly, but its implementation technique may contain weaknesses that could lead to failures.

Utilities are also available to aid the developer with data loading and capture. Data loading and capture allows the model, or portions of it, to be exercised with representative data. By simulating users, sensor input, or other types of data streams and test cases, the model is put into a simulated test environment.

Metrics utilities give the M&S Developer quantitative information about the code. Items such as execution speed and trends, user-defined application specific metrics, and SE standard metrics are reported. This information gives insight to the efficiency of the code, which could affect the usability of the model. In the case of a real-time model, code efficiency may be critical.

Commercial Verification Tools

This section lists a number of specific verification tools and products. These are intended only as examples of the types of verification tools available in the M&S community and should not be considered as endorsements of specific vendors.

Verification Tool Vendors

A best-of-class IDE will provide most, if not all, of the tools necessary to enable a smooth verification process for the M&S life cycle. The table below lists a few vendors that have products providing verification functions. In addition to a variety of vendors, there is variety in modeling and code generation languages. This variety is valuable, giving the developer a choice. However, if not properly managed, there can be as many tools and methods as there are developers.

Some Verification Tool Vendors			
Vendor	**Product**	**Web Address**	**Description**
Advanced Software Technologies	GDPro	www.advancedsw.com	Generates UML diagrams from code or code from UML diagrams. Reverse engineering. Supports C++ and Java.
Rational	Rational Rose 98i	www.rational.com	Code generation from UML and UML from code. Languages include C++, Java, and Visual Basic. Reverse engineering.
Mark V	ObjectMaker	www.markv.com	Supports over 30 analysis and design notations. Reverse engineering.
Platinum (now Computer Assoc.)	Paradigm Plus	www.platinum.com	Has impact analysis and traceability support features. Supports UML.
Aonix	Software Through Pictures	www.aonix.com	Visual modeling that supports UML, OMT, and Booch.
Object International	Together	www.oi.com	Supports UML / C++, and Java. Reverse Engineering.
Pragsoft	Pragmatica	www.pragsoft.com	Supports UML, Booch, Data Flow / C++, Java, IDL, RTF. Reverse Engineering.
MeteCASE	MetaEdit+	www.metacase.com	Supports UML, Booch, Rumbaugh, and more / Smalltalk, C++, Java, Delphi, SQL, IDL.
Select Software Tools	Select Enterprise	www.selectst.com	Supports UML / C++, Java, Forte, Visual Basic.
Visual Object Modelers	Visual UML	www.visualobjectmodelers.com	Allows user to create UML diagrams.
Popkin Software	System Architect 2001	www.popkin.com	Supports UML code generation to Java and C++.
Microtool	objectiF	www.microtool.de	Supports code generation, UML,

Some Verification Tool Vendors			
Vendor	**Product**	**Web Address**	**Description**
			etc.
Adaptive Arts	Simply Objects	www.adaptive-arts.com	Supports code generation and several modeling/programming languages.
Excel Software	WinA&D	www.excelsoftware.com	Uses verification reports to check consistency between class diagrams and project dictionary. Supports data modeling.
Blue River Software	V32	www.blue-river-software.com	Supports code generation from diagrams.
Project Technology	BridgePoint	www.projtech.com	Uses the Shlaer-Mellor method. Has a code verifier. Supports UML.
Project Technology	DesignPoint	www.projtech.com	Translates UML or SM models into source code for a variety of target platforms and languages.
Mega International	ISOA	www.mega.com	Supports UML. Has a code generator.
Innovative Software	Object Engineering Workbench	www.innovative-software.co.uk	Supports UML, Java, C++.
ObjectTime	ObjecTime Developer for C	www.objectime.com	For real-time systems.
Structured Technology Group	AxiomSys	www.stgcase.com	For small-to-medium projects. Can trace any type of information to the processes, modules and data items in the model. Validates trace files.
Structured Technology Group	AxiomDsn	www.stgcase.com	For medium-to-large projects. Allows user to build a detailed software model, trace when requirements are fulfilled, validate the model, etc.

M&S development tools should be selected carefully to ensure the free flow of information between the functions in the M&S development and V&V processes. The most conservative approach is to select a single vendor and methodology. However, one vendor may not provide all of the required utilities. While most IDE vendors provide a large selection of utilities within their own framework, give consideration to vendors that have an open architecture, permitting third-party add-ons.

Choosing multiple vendors is a viable option to exploit strengths of different products at specific points in the M&S life cycle. The flow of information between different tools needs to be laid out to ensure proper communications. For example, a database reverse engineering utility may interface effortlessly to some modeling tools, while being incompatible with others. Information standards are often in place to address

information exchange between tools, but it is likely that some situations will require specialized software development.

Connecting Verification Tools to the M&S Development Process

The Simulation Validation (SIMVAL) 99 Symposium, co-sponsored by the Military Operations Research Society (MORS) and the Society of Computer Simulation (SCS) International, focused on tools and technologies supporting VV&A [Glasow and Pace, 1999]. One of the SIMVAL working groups specifically examined the use of tools and techniques to support verification. The group concluded that

- computer-automated support tools are useful to support requirements verification
- the conceptual model requires tools that promote a standard approach for development
- existing tools are sufficient for the design and coding (implementation) phases of M&S development and verification [Glasow and Pace, 1999]

The table below re-lists the products presented in the verification tool vendor table, mapping their usage to particular phases of M&S development.

Product Application by M&S Development Phase				
Vendor	**M&S Requirements**	**Conceptual Model**	**M&S Implementation**	**M&S Application**
CodeSurfer			X	X
CIAO			X	X
GDPro		X	X	
Rational Rose 98i		X	X	
ObjectMaker		X	X	
Paradigm Plus	X	X	X	X
Software Through Pictures		X	X	
Together		X	X	
Pragmatica		X	X	
MetaEdit+		X	X	
Select Enterprise		X	X	
Visual UML		X	X	
System Architect 2001		X	X	
ObjectiF		X	X	
Simply Objects		X	X	

Product Application by M&S Development Phase				
Vendor	**M&S Requirements**	**Conceptual Model**	**M&S Implementation**	**M&S Application**
WinA&D		X	X	
V32		X	X	
BridgePoint		X	X	
DesignPoint		X	X	
ISOA		X	X	
Object Engineering Workbench		X	X	
ObjectTime Developer for C		X	X	
AxiomSys	X	X		X
AxiomDsn	X	X	X	X
Win/Xrunner	X			
ObjectGEODE	X	X	X	X
ReqisitePro	X			

Validation Tools

A second working group at SIMVAL 99 focused on the use of tools and technologies supporting validation. That working group observed that improvements are desperately needed for tools that support the development of model validation criteria and validation of the conceptual model. However, tools do exist to support data and results validation including database management systems, data modeling tools and data manipulation tools [Glasow and Pace, 1999]. In addition, a reasonable selection of 70-80 tools (actually slightly less due to rehosting or repackaging of applications to different names) is available if formal methods are considered as validation tools[1].

Tools applicable to formal methods are typically university development projects. Many of the tools can be downloaded from the WWW with no licensing. Some research teams offer well-developed packages with online users groups. In other cases, technical support consists of a single member of the development team. Formal method tools offer little with respect to user interface. The development efforts are mainly focused on functionality with minimal user interface development. There are a few exceptions, but the user interface is more of an after-thought than an integral part of

[1] Side-by-side testing of a model with a real-world system is adequate for result validation in most situations. If the model and its products are easily analyzed, or a subset of key scenarios is defined as the intended purpose, empirical analysis will likely be the most cost-effective approach. However, some models may involve so many permutations that empirical analysis becomes impractical. In these cases, the application of formal methods may aide the validation process.

the tool. Typically, because of this lack of a user interface and the nature of their use, formal methods require a high level of expertise to achieve proficiency.

The sections below discuss some basic validation tools. These fall into two categories, general purpose tools and formal methods, as listed below:

- general purpose tools supporting data and results validation
  - database management systems
  - data manipulation tools
  - data modeling tools
- formal methods
  - formal languages
  - mechanized reasoning tools
  - model checkers

General Purpose Tools

A number of different general purpose tools can be used to support data validation and results validation.

Database Management Systems (DBMSs) -- A DBMS is a software tool or collection of tools that is responsible for querying and modifying the contents of a database. Many database systems (such as Microsoft Access) come with a built-in DBMS. Most packages also allow the automatic generation of web pages that act as an online interface to the database so that it can be managed remotely. DBMS can be used to support data and results validation.

Data Manipulation Tools -- A number of commercial packages exist for general-purpose data manipulation, processing, and visualization (e.g., MATLAB by the Mathworks, PV-WAVE by Visual Numerics, Microsoft Excel). Such tools come with extensive libraries that support a large number of mathematical operations. With only a few lines of code it is possible to display data in multiple formats enabling data and results validation activities.

Data Modeling Tools -- Data modeling is the representation of data objects in a software system. It involves defining the relevant information structures in the system as well as specifying the relationships among them.

> Example:
>
> In designing an online credit-card payment system, a data object called CreditCard can be defined. This object can have several attributes, such as Number, ExpirationDate, and Owner. CreditCard can “belong to” another data object called

Customer, which has the attributes Name, Age, and Location. To prevent fraud, one function of the system may be to nullify the transaction if Customer.CreditCard.Owner does not equal Customer.Name.

Of course, data modeling is not absolutely necessary to develop such a system; however, it certainly helps to think of designing the system in terms of its functionality rather than in terms of code.

Formal Methods

Formal methods use formal logic to express a model's behavior. Formal methods are often applied in mission-critical, life-and-death modeling situations and are proving to be a powerful validation technique.

The purpose of formalizing a model is to express, in detail, what an algorithm does without the complexity of how it does it. By eliminating the details of implementation required by normal code development, algorithms and their relationships are isolated for evaluation.

Formal Languages -- A formal language is used to generate a formal description of a model either from the M&S requirements or directly from an M&S product. Formal notation goes beyond the simple process-relationship description to describe in detail what is performed within each function. This form of semi-automated desk checking is often adequate for uncovering algorithm errors or flaws in a conceptual model.

Mechanized Reasoning Tools (Provers) -- A mechanized reasoning tool (automated theorem prover) often supplements a formal language. Theorem provers (with formal languages) are typically university development programs that are continually evolving to include additions to the knowledge base. A prover's input language, a formal notation, can be used for formal specification alone or as a tool to break the model down into abstract objects of reasonable size for submission to the prover.

Automated theorem proving is an iterative process. The user must develop and submit a theorem, and any additional information about the theorem, to the prover. The prover applies rules of deduction and specific knowledge (if available) to the theorem to attempt a proof. It is up to the user to determine if the results are satisfactory. If not, information from the prover is used as additional knowledge and resubmitted until the proof is satisfied. A completed proof can be added to the knowledge base of the prover giving it additional knowledge to apply to future proofs.

Model Checkers -- A model checker is a utility that can be used in either of the V&V processes. If a model is characterized as a finite state machine, a model checker will exercise all available permutations of each transition in the model and their inter-relationships. As a validation tool, a conceptual model can be modeled as a state machine and examined for the purpose of proof of concept.

Commercial Validation Tools

This section lists a number of specific validation tools and products. These are intended only as examples of the types of validation tools available in the M&S community and should not be considered as endorsements of specific vendors.

Validation Tool Vendors

Software Validation Tools			
Tool	**Vendor**	**Description**	**Reference**
ObjectGEODE	CS Verilog	allow specification of how software is expected to behave	www.verilogusa.com
WinRunner/ Xrunner	Mercury Interactive	allow Developer to design test scripts that operate the application and can be replayed to validate functionality	www.merc-int.com

Data and Results Validation Tools		
DBMS	**Organization**	**Reference**
Database Management Systems (DBMS)		
Access	Microsoft	www.microsoft.com
FilemakerPro	FileMaker (formerly Claris)	www.filemaker.com
Oracle 8I	Oracle	www.oracle.com
DB2	IBM	www.software.ibm.com/data
Decision Frontier	Informix	www.informix.com
PostgreSQL	PostgreSQL Project	www.postgresql.org
Adaptive Server	Sybase	www.sybase.com
InterBase	InterBase (part of Inprise/Borland)	www.interbase.com
FirstSQL	FirstSQL	www.firstsql.com
MySQL	T.c.X	www.tcs.se
Data Manipulation Tools		
MATLAB	The Mathworks	www.matlab.com
Excel	Microsoft	www.microsoft.com
PV-WAVE	Visual Numerics	www.vni.com
Data Modeling Tools		
Erwin	Logic Works (now Computer Assoc.)	www.logicworks.com
Visible Advantage	Visible	www.visible.com
Database Builder	Mega International	www.mega.com

Formal Method Tool Vendors

Because formal methods are inherently isolated processes, integration with M&S tools is not as important. Information extracted from the M&S specification or product is needed to support formal methods, but formal methods typically do not produce a product that is directly incorporated into the M&S development process. If an algorithm or a part of a model presents an unmanageable number of permutations or is deemed mission critical, formal methods may be useful. The table below includes a sampling of tools used in formal methods.

Sample Formal Validation Tools		
Tool	**Organization**	**Description**
EVES	ORA Canada www.ora.on.ca	Uses a formal notation called Verdi and a automatic deduction system called NEVER
Higher Order Logic (HOL)	University of Cambridge www.cl.cam.ac.uk/Research/HVG/HOL/	Uses a Standard Meta Language (ML) as its notation
Larch / Larch Prover (LP)	MIT www.sds.lcs.mit.edu/spd/larch/	interactive proving system for multisorted first-order logic
Nqthm	University of Texas - Austin www.cs.utexas.edu/users/boyer/ftp/nqthm/	Boyer-Moore prover
Nuprl	Cornell www.cs.cornell.edu/Info/Projects/NuPrl/	based on Proof Refinement Logic
Prototype Verification System (PVS)	SRI pvs.csl.sri.com	specification language is based on higher-order logic; prover is a collection of inference procedures
Rigorous Approach to Industrial SE (RAISE)	University of Edinburgh dream.dai.ed.ac.uk/raise/	RSL specification language; specifically intended for software systems
Vienna Development Method (VDM)	IFAD www.ifad.dk/vdm/	specification language is VDM-SL. Intended for software systems
Z (pronounced "zed")	Oxford University www.comlab.ox.ac.uk/archive/z.html	a specification language based on first-order predicate logic
COrdination Specification Analysis (COSPAN)	Bell Labs netlib.bell-labs.com/cm/cs/what/formal_methods	based on the S/R language. Is a model checking system for other systems
Murphi	Stanford sprout.stanford.edu/dill/murphi.html	specification language based on a set of action rules which execute repeatedly in an infinite loop

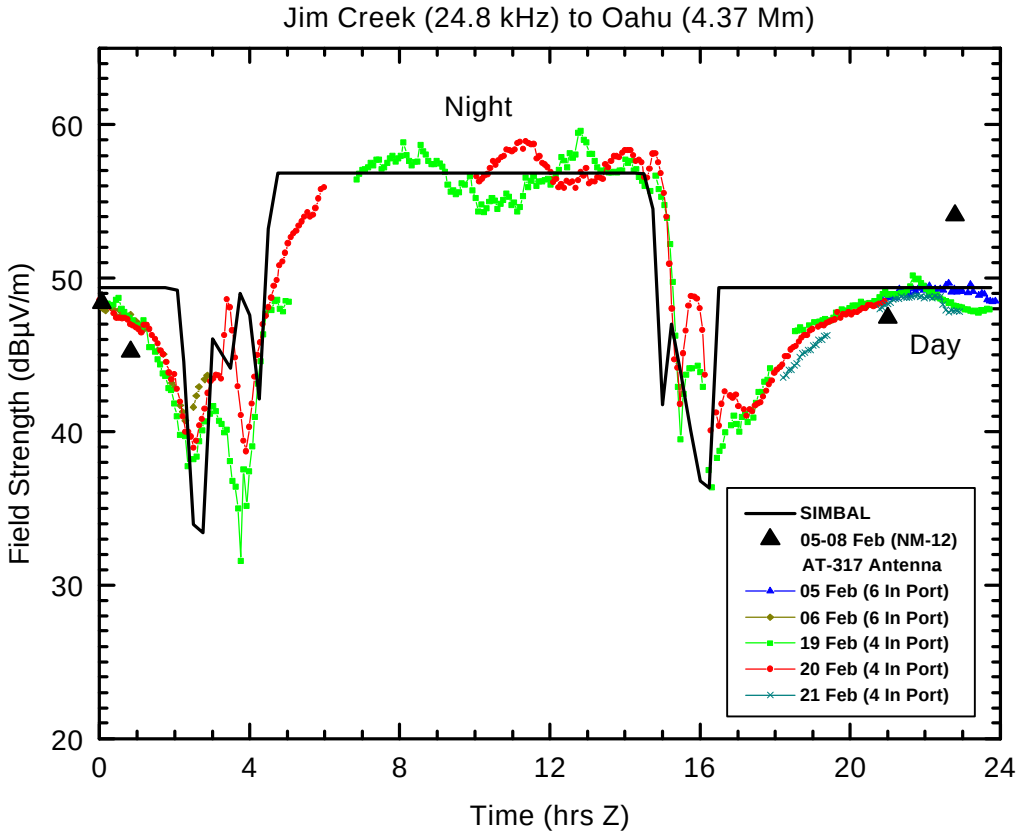
Sample Formal Validation Tools		
Tool	**Organization**	**Description**
Symbolic Model Verifier (SMV)	Carnegie Mellon University www.cs.cmu.edu/~modelcheck	checks finite state systems against their specifications
Interactive Mathematical Proof System (IMPS)	Harvard FTP://math.harvard.edu/imps/	consists of a database of mathematics and tools for exploring, extending, and communicating its contents
Spin	Bells Labs netlib.bell-labs.com/netlib/spin/	User uses PROMELA (PROcess MEta Language to define a formal model and employs SPIN to check and trace logical errors
Unity Verifier	University of Texas - Austin www.cs.utexas.edu/users/psp/	both a programming notation and a logic to reason about parallel and distributed programs
Failures-Divergence Refinement (FDR, FDR2)	Formal Systems www.formal.demon.co.uk	based on CSP. allows verification of finite-state systems; helps investigate systems which fail checks
ProBE	Formal Systems www.formal.demon.co.uk	based on CSP. shows how a CSP process evolves as User chooses among available actions
Petri Nets Tools	Various Institutions home.arcor-online.de/wolf.garbe/petrisurv2.html	used to model concurrent systems; the first general theory for parallel systems
B-Toolkit/B-Method	B-core www.b-core.com	a collection of techniques for the specification, design and implementation of software components
Isabelle	University of Cambridge/TU Munich isabelle.in.tum.de/	a generic theorem proving environment
VeriSoft	Bell Labs www.bell-labs.com/projects/verisoft/	uses systematic state-space exploration to locate deadlocks, assertion violations, and other conditions

Validation Example

The Strategic Communications Continuing Assessment Program (SCAP) has been in existence since 1979 and provides the Navy with predictions of connectivity to U.S. strategic forces under stressed conditions. The SCAP model is comprised of a nuclear propagation data generation model (Simulation of Multiple Bursts and Links [SIMBAL] built and maintained by Kaman Sciences) and a communications network simulator (Navy Strategic Communications Simulator [NSCS] built and maintained by The Johns

Hopkins University Applied Physics Laboratory). SCAP provides the Navy with predictions of Probability of Correct Message Receipt (PCMR) and Times of Receipt (TORs) into the Atlantic and Pacific Oceans. Over the years, the SCAP model has undergone extensive results validation. Some of those efforts and findings are summarized here.

SIMBAL provides signal level predictions as a function of transmitter frequency, path length, transmitter/receiver height, time of day, season, ground conductivity, and antenna polarization. Predictions are made based on pre-computed databases at selected frequencies; SIMBAL is primarily used to compute signal levels on large networks and the use of pre-computed databases reduces model runtimes. However, when predictions are needed at frequencies other than those in the pre-computed databases, prediction accuracy may be reduced. JHU/APL and Kaman Sciences have examined SIMBAL predictions extensively to understand the fidelity and accuracy of model predictions. The figure below compares model output on predictions during daytime, nighttime, and transition with collected data on very low frequency (VLF) transmissions from Jim Creek [VLF transmitter] to Oahu.



Comparison of Model Output to Collected Data

The table below summarizes comparisons of this type, collected over many years, on a variety of transmitter/receiver pairs. It shows average error and standard deviation. The tool used to support these comparisons was Origin.

SIMBAL VLF/LF Predicted Signal Levels vs. Measured Data			
Condition	**Number of Samples**	**Average Error (dB)**	**Standard Deviation (dB)**
Mid-Latitude	147	-0.8	5.2
VLF	27	0.4	3.8
LF	114	-0.9	5.4
VLF Day	15	-0.7	3.1
VLF Night	12	1.8	4.4
VLF Transition	26	1.5	7.0

References

Glasow, P., Pace, D., “SIMVAL 99 -- Making VV&A Effective and Affordable,” *Phalanx*, March 1999, p. 22-24.

§ § § § § § §